# Handling an HTTP redirect with QNetworkAccessManager

## Overview

This code shows how to handle an HTTP redirect with `QNetworkAccessManager`.

## Header file

```
/*
 * Copyright (c) 2009 Nokia Corporation
 */

#ifndef QNAMREDIRECT_H
#define QNAMREDIRECT_H

#include <QtGui/QMainWindow>
#include <QtGui/QFrame>
#include <QtGui/QVBoxLayout>
#include <QtGui/QLabel>
#include <QtGui/QPushButton>

#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>

#include <QtCore/QPointer>
#include <QtCore/QUrl>

class QNAMRedirect : public QMainWindow
{
    Q_OBJECT

public:
 QNAMRedirect(QWidget *parent = 0);
 ~QNAMRedirect();

private slots:
 void doRequest();
 void replyFinished(QNetworkReply* reply);

private:
 QPointer<QNetworkAccessManager> _qnam;
 QUrl _originalUrl;
 QUrl _urlRedirectedTo;

 QPointer<QLabel> _textContainer;

 QNetworkAccessManager* createQNAM();
 QUrl redirectUrl(const QUrl& possibleRedirectUrl,
                  const QUrl& oldRedirectUrl) const;
};

#endif // QNAMREDIRECT_H
```


(C) Copyright Nokia 2014. All rights reserved.

## Source file

```
/*
 * Copyright (c) 2009 Nokia Corporation
 */

#include "qnamredirect.h"

QNAMRedirect::QNAMRedirect(QWidget *parent) : QMainWindow(parent) {
 _qnam = createQNAM();
 _originalUrl = "http://www.wikipedia.org/wiki/URL_redirection";

 QVBoxLayout* layout = new QVBoxLayout;

#ifdef Q_OS_SYMBIAN
 setStyleSheet("QLabel, QPushButton{ font: 5pt; }");
#endif

 _textContainer = new QLabel("Click the button to test URL redirect!");
 _textContainer->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);
 _textContainer->setAlignment(Qt::AlignCenter);
 _textContainer->setWordWrap(true);
 layout->addWidget(_textContainer);

 QPushButton* testButton = new QPushButton("Click here!");
 /* If the button is clicked, we'll do a request. */
 connect(testButton, SIGNAL(clicked()),
         this, SLOT(doRequest()));
 layout->addWidget(testButton);

 QFrame* frame = new QFrame;
 frame->setLayout(layout);
 setCentralWidget(frame);
}

QNAMRedirect::~QNAMRedirect() {
 if(_qnam) {
  _qnam->deleteLater();
 }
 if(_textContainer) {
  _textContainer->deleteLater();
 }
}

QNetworkAccessManager* QNAMRedirect::createQNAM() {
 QNetworkAccessManager* qnam = new QNetworkAccessManager(this);
 /* We'll handle the finished reply in replyFinished */
 connect(qnam, SIGNAL(finished(QNetworkReply*)),
         this, SLOT(replyFinished(QNetworkReply*)));
 return qnam;
}

void QNAMRedirect::doRequest() {
 QString text = "QNAMRedirect::doRequest doing request to ";
 text.append(this->_originalUrl.toString());
 this->_textContainer->setText(text);
```


(C) Copyright Nokia 2014. All rights reserved.

```
 /* Let's just create network request for this predifned URL... */
 QNetworkRequest request(this->_originalUrl);
 /* ...and ask the manager to do the request. */
 this->_qnam->get(request);
}

void QNAMRedirect::replyFinished(QNetworkReply* reply) {
 /*
  * Reply is finished!
  * We'll ask for the reply about the Redirection attribute
  * http://doc.trolltech.com/qnetworkrequest.html#Attribute-enum
  */
 QVariant possibleRedirectUrl =
          reply->attribute(QNetworkRequest::RedirectionTargetAttribute);

 /* We'll deduct if the redirection is valid in the redirectUrl function */
 _urlRedirectedTo = this->redirectUrl(possibleRedirectUrl.toUrl(),
                                      _urlRedirectedTo);

 /* If the URL is not empty, we're being redirected. */
 if(!_urlRedirectedTo.isEmpty()) {
  QString text = QString("QNAMRedirect::replyFinished: Redirected to ")
                        .append(_urlRedirectedTo.toString());
  this->_textContainer->setText(text);

  /* We'll do another request to the redirection url. */
  this->_qnam->get(QNetworkRequest(_urlRedirectedTo));
 }
 else {
  /*
   * We weren't redirected anymore
   * so we arrived to the final destination...
   */
  QString text = QString("QNAMRedirect::replyFinished: Arrived to ")
                        .append(reply->url().toString());
  this->_textContainer->setText(text);
  /* ...so this can be cleared. */
  _urlRedirectedTo.clear();
 }
 /* Clean up. */
 reply->deleteLater();
}

QUrl QNAMRedirect::redirectUrl(const QUrl& possibleRedirectUrl,
                               const QUrl& oldRedirectUrl) const {
 QUrl redirectUrl;
 /*
  * Check if the URL is empty and
  * that we aren't being fooled into a infinite redirect loop.
  * We could also keep track of how many redirects we have been to
  * and set a limit to it, but we'll leave that to you.
  */
 if(!possibleRedirectUrl.isEmpty() &&
    possibleRedirectUrl != oldRedirectUrl) {
  redirectUrl = possibleRedirectUrl;
 }
 return redirectUrl;
```


(C) Copyright Nokia 2014. All rights reserved.

```
}
```

## Postconditions

You are able to handle HTTP redirects with `QNetworkAccessManager`.

## Supplementary material

- You can test `QNAMRedirect` with a test application. The application is available for download at Media:QNAMRedirect.zip.

## See also

- Creating an HTTP network request in Qt
- QNetworkAccessManager documentation
- QNetworkRequest documentation


(C) Copyright Nokia 2014. All rights reserved.